Chiang Mai J. Sci. 2009; 36(1) : 9-23
www.science.cmu.ac.th/journal-science/josci.html
Contributed Paper

Solving First Order Systems of Ordinary Differential Equations Using Parallel R-Point Block Method of Variable Step Size and Order

Zurni B. Omar* [a], and Mohamed Suleiman [b]**
[a] UUM College of Arts and Sciences, Quantitative Science Building, Universiti Utara Malaysia, 06010 UUM Sintok, Kedah Darul Aman, Malaysia.
[b] Department of Mathematics, Universiti Putra Malaysia, 43400 UPM, Serdang Selangor, Malaysia.
Author for correspondence; *e-mail: zurni@uum.edu.my **e-mail: m1suleiman@yahoo.com.my

Recieved : 6 October 2008
Accepted : 14 November 2008.

ABSTRACT

A new parallel method for solving first order systems of ordinary differential equations using variable step size and order (back value) is developed. The method, known as parallel R-point block method, calculates the numerical solution at more than one point simultaneously.

The program of the method employed is run on a shared memory Sequent Symmetry S27 parallel computer. Computational advantages are presented comparing the results obtained by the new method with that of conventional 1-point method. The numerical results show that the new method reduces the total number of steps and execution time. It is also discovered that the accuracy of the parallel r-point block method and the 1-point method is comparable particularly when finer tolerances are used.

Keywords: ordinary differential equations, first order systems, parallel, r-point block method, variable step size and order.

1. INTRODUCTION

Numerous methods for solving first order systems of ordinary differential equations (ODEs) have been discussed by several researchers such as Henrici [1] and Lambert [2]. These methods, however, approximate solution at one point at a time and use constant step size and order.

The objective of this paper is to develop a method that approximates solutions of non-stiff equations for first order system at *r* points simultaneously using variable step size and order. The idea is the extension of the previous work conducted by Omar and Suleiman [3]. We will apply some of the techniques proposed by Suleiman [4] for first order systems.

The general system of first order ODEs is in the following form

$$y_i' = f_i(x, \tilde{Y}), \quad i = 1,2,.....,s \tag{1.1}$$

with

$$\tilde{Y}(a) = \eta \text{ for } a \le x \le b,$$
$$\tilde{Y}^T(x) = (y_1, y_2, ..., y_s), \text{ and}$$
$$\tilde{\eta}^T(x) = (\eta_1, \eta_2, ..., \eta_s)$$

where T denotes transpose. For the purpose of simplicity of description and without loss of generality, we will focus our discussion on a single equation

$$y' = f(x, y), \quad y(a) = \eta, \quad a \le x \le b \tag{1.2}$$

2. ALGORITHMS TO GENERATE THE INTEGRATION COEFFICIENTS

Let us define the interpolating polynomial $p_{k,n}(x)$ as follows

$$P_{k,n}(x) = f_n + (x - x_n) f_{[n,n-1]} + ... + (x - x_n)....(x - x_{n-k+2}) f_{[n,n-1,....,n-k+1]} \tag{2.1}$$

where

$$f_{[n,n-1,....,n-i]} = \frac{f_{[n,n-1,.....,n-i+1]} - f_{[n-1,n-2,....,n-i]}}{x_n - x_{n-i}} \tag{2.2}$$

Now, define

$$g_{i,1}^{(d)}(x) = \int_{x_n}^{x_{n+d}} p_{n,i}(x)\, dx$$
$$g_{i,0}^{(d)} = p_{n,i}(x_{n+d}) \quad \text{for d} = 1,2,....,\text{R}, \; i = 1,2,....,k \tag{2.3}$$

where

$$p_{n,i}(x) = (x - x_n)(x - x_{n-1})....(x - x_{n-i+1}) \quad \text{for } i = 1,2,....,k, \text{ and} \tag{2.4}$$
$$p_{n,0}(x) = 1$$

Substituting (2.4) in (2.3) gives

$$g_{i,1}^{(d)}(x) = \int_{x}^{x_{n+d}} (x - x_n)(x - x_{n-1})....(x - x_{n-i+1})\, dx \tag{2.5}$$

Let $u = x - x_{n-i+1}$, $dv = (x - x_n)....(x - x_{n-i+2})$. Solving (2.5) using integration by parts yields

$$g_{i,1}^{(d)}(x) = \int_{x}^{x_{n+d}} (x - x_n)(x - x_{n-1})....(x - x_{n-i+1})\, dx$$

$$- \int_{x_n}^{x_{n+d}} \int_{x_n}^{x} (x - x_n)....(x - x_{n-i+2}) dx dx \tag{2.6}$$

It follows that

$$g_{i,1}^{(d)} = (x_{n+d} - x_{n-i+1}) g_{i-1,1}^{(d)} - g_{i-1,2}^{(d)} \tag{2.7}$$

For $i = 0$,

$$g_{0,1}^{(d)} = \int_{x_n}^{x_{n+d}} dx = \frac{dh_{n+1}}{t!} \tag{2.8}$$

where $h_r = x_r - x_{r-1}, \quad r = 1,2,.....$

The number of coefficients required depends on the order of the differential equation, that is 1, and also the order of the method being used in solving the problem. Define $G_{\max} = \max(k_i + 1)$ where k_i is the current k_i - step method used for each equation in the system. Let $k_{\max}$ be the maximum value of k_i for the system. We will calculate the values of $k_{i,t}$ for $i = 1,2,...,k_{\max}+1$ and t = 0,1,...,$G_{\max}+1$.

3. DERIVATION OF THE INTEGRATION FORMULAE

In this section we will derive integration formulae based on the number of back values of y' stored. Since a predictor-corrector scheme will be used, it is appropriate to describe the details of prediction first and then followed by correction.

3.1 Prediction

Integrating (1.2) once leads to

$$y(x_{n+d}) = y(x_n) + \int_{x_n}^{x_{n+d}} f(x, \tilde{Y}(x)) dx....dx \tag{3.1}$$

The value $y(k_{n+d})$ can be approximated by replacing f with an interpolation polynomial $p_{k,n}(x)$ of degree k-1, which interpolates f at the preceding k mesh points, $x_n, x_{n-1}, ..., x_{n-k+1}$. The polynomial can be expressed in terms of divided differences as follows

$$P_{k,n}(x) = f_{[n]} + (x - x_n) f_{[n,n-1]} + ... + (x - x_n)....(x - x_{n-k+2}) f_{[n,n-1,....,n-k+1]}$$

Replace (3.1) in discrete form

$$y_{n+d} = y_n + \int_{x_n}^{x} P_{k,n}(x) dx \tag{3.2}$$

which is equivalent to

$$y_{n+d} = y_n + \sum_{i=0}^{k-1} g_{i,1} f_{[n,n-1,.....,n-i]} \tag{3.3}$$

Thus, the predictor formulae are

$$\begin{aligned} p_{n+d} &= y_n + \sum_{i=0}^{k-1} g_{i,1} f_{[n,n-1,....,n-i]} \\ p'_{n+d} &= \sum_{i=0}^{k-1} g_{i,0} f_{[n,n-1,....,n-i]} \end{aligned} \tag{3.4}$$

3.2 Correction

The corrector formulae is constructed to provide values that satisfy

$$y' = f(x_{n+d}, y_{n+d}) \tag{3.5}$$

However, the formulae are not exactly satisfied since a fixed number of predictor-corrector iterations is used.

Let ${}^s y_{n+d}$ denote the s th iterative value of y_{n+d} and ${}^s e_d = {}^s y_{n+d} - {}^{s-1} y_{n+d}$ the difference between the s th and s-1 th iterative values of y_{n+d}. For the case $s = 1$, ${}^1 e_d$ is simply written as e_d.

Let ${}^0 y_{n+d} = p_{n+d}$. If standard PECE method is employed, we perform the first evaluation after getting ${}^0 y_{n+d}$

$${}^1 y' = f(x_{n+d}, {}^0 y_{n+d}) = {}^1 f_{n+d} \tag{3.6}$$

Now, let $p_{k+1,n+d}(x)$ be the interpolation polynomial of degree k which interpolates f at $\{(x_{n+d}, {}^1 f_{n+d}), (x_n, f_n), ..., (x_{n-k+1}, f_{n-k+1})\}$. $p_{k+1,n+d}(x)$ can be written in terms of $p_{k,n}(x)$ as follows

$$\begin{aligned} P_{k+1,n+d}(x) &= f_{[n]} + (x - x_n) f_{[n,n-1]} + (x - x_n)(x - x_{n-1}) f_{[n,n-1,n-2]} + ... \\ &\quad + (x - x_n)(x - x_{n-1})....(x - x_{n-k+2}) f_{[n,n-1,....,n-k+1]} \\ &\quad + (x - x_n)(x - x_{n-1})....(x - x_{n-k+2})(x - x_{n-k+1}) f_{[n,n-1,....,n-k+1,n+d]} \\ &= P_{k,n}(x) + (x - x_n)(x - x_{n-1})....(x - x_{n-k+2})(x - x_{n-k+1}) f_{[n+d,n,n-1,....,n-k+1]} \end{aligned}$$

Integrating the polynomial $p_{k+1,n+d}(x)$ t times between the limits x_n and x_{n+d} leads to

$$\begin{aligned} {}^1 y_{n+d} &= p_{n+d} + g^{(d)}_{k,1} f_{[n+d,n,n-1,.....,n-k+1]} \\ {}^1 y'_{n+d} &= p'_{n+d} + g^{(d)}_{k,0} f_{[n+d,n,n-1,.....,n-k+1]} \end{aligned} \tag{3.7}$$

Let $e_d = {}^1 y'_{n+d} - p'_{n+d}$. This implies that

$$e_d = f(x_{n+d}, \tilde{P}_{n+d}) - p'_{n+d} \tag{3.8}$$

Thus the corrected values are

$$\begin{aligned} {}^{1}y_{n+d} &= p_{n+d} + \frac{g_{k,t}^{(d)}}{g_{k,0}^{(d)}} e_d \\ {}^{1}y'_{n+d} &= p'_{n+d} + e_d \end{aligned} \tag{3.9}$$

Finally the final evaluation is performed, that is

$$y'_{n+d} = f(x_{n+d}, {}^{1}y'_{n+d}) \tag{3.10}$$

The process of PECE method is now completed.

Suppose a PECECE method is employed instead. Substitute y'_{n+d} with ${}^{2}y_{n+d}$ in (3.10) we have

$${}^{2}y_{n+d} = f(x_{n+d}, {}^{1}y_{n+d}) = {}^{2}f_{n+d} \tag{3.11}$$

Let $p_{k+1,n+d}(x)$ interpolates f at $\{(x)_{n+d}, {}^{2}f_{n+d}), (x_n, f_n, ..., (x_{n-k+1}, f_{n-k+1})\}$.
Writing $p_{k+1,n+d}(x)$ in terms of $p_{k,n}(x)$ gives

$$p_{k+1,n+d}(x) = p_{k,n}(x) + (x-x_n)(x-x_{n-1})....(x-x_{n-k+2})(x-x_{n-k+1}) {}^{2}f_{[n+d,n,n-1,....,n-k+1]}$$

We then integrate $p_{k+1,n+d}(x)$ t times between x_n and x_{n+d} to get

$$\begin{aligned} {}^{2}y_{n+d} &= p_{n+d} + g_{k,1}^{(d)} {}^{2}f_{[n+d,n,n-1,....,n-k+1]} \\ {}^{2}y'_{n+d} &= p'_{n+d} + g_{k,0}^{(d)} {}^{2}f_{[n+d,n,n-1,....,n-k+1]} \end{aligned} \tag{3.12}$$

By letting ${}^{2}e_d = {}^{2}y'_{n+d} - p'_{n+d}$ we thus get

$${}^{2}e_d = f(x, {}^{1}y_{n+d} - p'_{n+d} \tag{3.13}$$

This also implies

$$f_{[n+d,n,n-1,....,n-k+1]} = \frac{{}^{2}e_d}{g_{k,0}^{(d)}} \tag{3.14}$$

Hence, the corrected values are

$$\begin{aligned} {}^{2}y_{n+d} &= p_{n+d} + \frac{g_{k,t}^{(d)}}{g_{k,0}^{(d)}} {}^{2}e_d, \quad t = 1,2,....,d \\ {}^{2}y'_{n+d} &= p'_{n+d} + {}^{2}e_d \end{aligned} \tag{3.15}$$

The final evaluation

$$y'_{n+d} = f(x_{n+d}, {}^{2}\tilde{Y}_{n+1})$$

concludes the PECECE mode.

4. ESTIMATING THE ERROR

The estimation for the local errors on each step of the integration will be investigated in this section. The local error is given by

$$E_{d,k} = u_n(x_{n+d}) - y_{n+d} \tag{4.1}$$

where $u_n(x)$ is the solution of $u_n' = f(x,u_n)$ and $u_n(x_n) = y_n$.

The results of formulae of different orders are compared to get the estimation of the local errors. Let $y_{n+d}(k)$ and $y_{n+d}(k+1)$ be the results of stepping to x_{n+d} using k-step and (k+1) step methods respectively. For a constant step size, the principal term of the local error is correctly estimated asymptotically by

$$\begin{aligned} y_{n+d}(k+1) - y_{n+d}(k) &= [u_n(x_{n+d}) - y_{n+d}(k)] - [u_n(x_{n+d}) - y_{n+d}(k+1)] \\ &\approx u_n(x_{n+d}) - y_{n+d}(k) \end{aligned} \tag{4.2}$$

Using a similar approach of (3.13), we can estimate the local error $E_{d,k}$ of the less accurate result given by

$$E_{d,k} \approx y_{n+d}(k+1) - y_{n+d}(k) \tag{4.3}$$

where Let $y_{n+d}(k)$ and $y_{n+d}(k+1)$ are the results of using the iterative mode PEC_kE and $PEC_{k+1}E$ respectively. Let $E_{d,k} = y_{n+d}(k+1) - y_{n+d}(k)$ denotes the estimated error in $y_{n+d}(k)$ at x_{n+d}. It follows from (3.9) that

$$y_{n+d}(k) = p_{n+d} + \frac{g_{k-1,1}^{(d)}}{g_{k-1,0}^{(d)}} + \bar{e}_d$$

$$y_{n+d}(k+1) = p_{n+d} + \frac{g_{k,1}^{(d)}}{g_{k,0}^{(d)}} \hat{e}_d$$

where $\bar{e}_d = y_{n+d}^{(d)}(k) - p_{n+d}^{(d)}$ and $\hat{e}_d = y_{n+d}^{(d)}(k+1) - p_{n+d}^{(d)}$.

As a result, we have

$$E_{d.k} = \frac{g_{k,1}^{(d)}}{g_{k,0}^{(d)}} \hat{e}_d - \frac{g_{k-1,1}^{(d)}}{g_{k-1,0}^{(d)}} \bar{e}_d \tag{4.4}$$

Now,

$$\begin{aligned} \bar{e}_d &= g_{k-1,0}^{(d)}(x_{n+d} - x_{n-k+1}) f_{[n+d,n,\ldots,n-k+1]} \\ &= g_{k,0}^{(d)} f_{[n+d,n,\ldots,n-k+1]} \\ &= \hat{e}_d \\ &= e_d \end{aligned} \tag{4.5}$$

Substituting the result of (4.5) in (4.4) gives

$$E_{d,k} = \left\{ g_{k,1}^{(d)} - (x_{n+d} - x_{n-k+1}) g_{k-1,1}^{(d)} \right\} f_{[n+d,n,.....,n-k+1]} \tag{4.6}$$

Applying (2.7) in (4.6) gives

$$E_{d,k} = -\frac{g_{k-1,2}^{(d)}}{g_{k,0}^{(d)}} f_{[n+d,n,....,n-k+1]}$$

which can be written in terms of e_d as

$$E_{d,k} = -\frac{dg_{k-1,2}^{(d)}}{g_{k,0}^{(d)}} e_d \tag{4.7}$$

The values of $E_{d,k-2}$, $E_{d,k-1}$ and $E_{d,k+1}$ which represent the local errors that would have been made had the step been taken with predictor and corrector of order $k-2$, $k-1$ and $k+1$ respectively are also needed. Those values are used for the order and step size strategy purposes. The approximation of the error estimate $E_{d,k-2}$ is given by

$$\tilde{E}_{d,k-2} = -g_{k-3,1}^{(d)} f_{[n+d,n,....,n-k+3]} \tag{4.8}$$

In a similar manner, $\tilde{E}_{d,k-1}$ and $\tilde{E}_{d,k+1}$ can be obtained by replacing k-2 in (4.8) with k-1 and k+1 respectively.

An issue arises here is to choose at which point of r-point method should the local error be controlled. It has been decided to just control the error at the first point, that is, $E_{1,k}$ since the order of the error is the same in h. Furthermore, the empirical evidence also has shown that it is more efficient to control the error at the first point and it was proven by the numerical results. For simplicity of notation, from now on, $E_{1,k}$ will be written as E_k.

5. ORDER AND STEP SIZE SELECTION

After performing integration steps at R points simultaneously, the decision whether or not to accept those results has to be made. The results are accepted if the local error at the first point satisfies the local accuracy requirements as follows

$$E_k < TOL * (A + B * p_{n+1})$$

with A=1. B=0 gives the absolute error test, A=0. B=1 a relative error test and A=B=1 a mixed error test. Then, regardless whether the results are accepted or not, the next step size and order have to be determined. We can do the correction, update new divided difference and proceed to the next integration steps if the results are accepted. Otherwise, go back and perform the integration steps again until the results are accepted. The process repeats until the end point is reached.

The same strategies as used by Suleiman [4] are applied for choosing the step size and order. The order (the number of back values) is lowered by one if we have the following cases

i) for k>2, $\max\left(\left|\widetilde{E}_{k-1}\right|,\left|\widetilde{E}_{k-2}\right|\right)\le\left|E_k\right|$ and for k=2, $\left|\widetilde{E}_{k-1}\right|\le 0.5\left|E_k\right|$.

ii) if k>1 and $\left|\widetilde{E}_{k-1}\right|\le\min\left(\left|E_k\right|,\left|\widetilde{E}_{k+1}\right|\right)$ when $\left|\widetilde{E}_{k+1}\right|$ is available.

On the other hand, we increase the order by one if after k+1 successful step at constant step size

i) $\left|\widetilde{E}_{k+1}\right|<\left|E_k\right|<\max(\left|\widetilde{E}_{k-1}\right|,\left|\widetilde{E}_{k-2}\right|$ for $k>1$

ii) $\left|\widetilde{E}_{k-1}\right|<0.5\left|E_k\right|$ for $k=1$

The value of k is restricted in the range $1\le k\le 12$. After choosing the order to be used, we rename it and the corresponding error estimate as E_k.

For the next integration steps, the estimated optimal step size, h_{new} is given by

$$h_{new}\approx TOL*\left(A+B\left|y_{n+1}\right|\right)$$

Let $\frac{1}{\theta_{n+1}}=A+B\left|y_{n+1}\right|$. Since E_k is $O(h^{k+p})$, we have

$$\theta_{n+1}\left|E_k(h)\right|=Ch^{k+p}$$

which implies

$$\theta_{n+1}\left|E_k(h_{new})\right|=Ch_{new}^{k+p}=TOL \tag{5.1}$$

This leads to

$$h_{new}=h*R \text{ where } R=\left(\frac{TOL}{\theta_{n+1}\left|E_k\right|}\right)^{\frac{1}{k+p}}$$

In practice, a safety factor, 0.8 for our code, is multiplied on the right hand side of (5.1) to give a more conservative estimate for h_{new} and thus reduce the number of steps rejected.

The step size algorithm when integration steps are successful are given by

$$h_{new}=\begin{cases}2h & \text{if } R^*\ge 2 \text{ and } R^*\le\sqrt{TOL}\\ h & \text{if } R^*\ge 2 \text{ and } R^*>\sqrt{TOL}\\ Rh & \text{if } 1.6<R^*<2 \text{ or } 0.5\le R^*<0.9\\ 0.5h & \text{if } R^*<0.5\end{cases} \tag{5.2}$$

where $R^*=0.8R$.

Meanwhile, the algorithm when a step failure occurs is

$$h_{new}=Rh \tag{5.3}$$

where the value R is 0.4 for the initial step and is 0.5 otherwise. When four repeated failures occur, the step size in (5.3) is further reduced by a factor of 10.

6. TEST PROBLEMS

The following problems were solved numerically using the 2-point (2PBVSO) and 1-point (1PVSO) block methods of variable step size and order:

Problem 1: $y_1' = 1 + y_1 + 2y_2,\ y_1(0) = 1,$
$y_2' = -1 + 4y_1 + 3y_2,\ y_2(0) = 2,$
$0 \le x \le 100.$

Solution: $y_1(x) = e^{5x} - e^{-x} + 1,$
$y_2(x) = 2e^{5x} + e^{-x} - 1.$

Source: Bronson [5].

Problem 2: $y_1' = y_2,\ y_1(0) = 0,$
$y_2' = -2y_2 - 5y_3 + 3,\ y_2(0) = 0,$
$y_3' = y_2 + 2y_3,\ y_3(0) = 1,$
$0 \le x \le 4\pi.$

Solution: $y_1(x) = 2\cos x + 6\sin x - 6x - 2,$
$y_2(x) = -2\sin x + 6\cos x - 6,$
$y_3(x) = 2\sin x - 2\cos x + 3.$

Source: Suleiman [6].

Problem 3: $y_1' = y_2,\ y_1(0) = 0,$
$y_2' = 2y_2 - y_1,\ y_2(0) = 1,$
$0 \le x \le 50.$

Solution: $y_1(x) = xe^x,$
$y_2(x) = (1 + x)e^x$

Source: Krogh [7].

7. NUMERICAL RESULTS

The tables below give the numerical results for Problems 1 to 3. The notations and abbreviations opted take the following meaning:

TIME	The execution time in microseconds needed to complete the integration in a given interval.
TOL	The chosen tolerance.
MAXE	The maximum error.
AVEERR	The average error.
SUSTEP	The number of successful steps.
FSTEPS	The number of step failures.
STEPS	The total number of steps.
S2PBVSO	The sequential implementation of the 2PBVSO method.
P2PBVSO	The parallel implementation of the 2PBVSO method.
R.STEP	The ratio step of the 2PBVSO method to the 1PVSO method.
R.TIME	The ratio time of the 2PBVSO method to the 1PVSO method.
MTD	The method employed.

The errors calculated are defined as

$$(e_i)_t = \left| \frac{(y_i)_t - (y(x_i))_t}{A + B(y(x_i))_t} \right|$$

where $(y)_t$ is the t-th component of $\tilde{y}$. $A = 1$, $B = 0$ corresponds to the absolute error test, $A = 1$, $B = 1$ corresponds to the mixed test and finally $A = 0$, $B = 1$ corresponds to the relative error test. For Problems 1-3, the mixed test is used. The maximum error and average error are given by

$$\text{MAXE} = \max_{1 \le i \le \text{SUSTEP}} \left(\max_{1 \le i \le s} (e_i)_t \right) \text{ and}$$

$$\text{AVEERR} = \frac{\sum_{i=1}^{\text{SUSTEP}} \sum_{t=1}^{s} (e_i)_t}{(R)(s)(\text{SUSTEP})}$$

where s is the number of the equations in the system and R is the number of point.

Tables 1-3 give the performance comparison between the parallel and sequential implementations of the 2PBVSO with the 1PVSO for solving the given problems. The measured parameters are the number of steps, maximum error, average error, failure steps and execution time. The efficiency in terms of the total number of steps and times for both sequential and parallel implementation of the 2PBVSO to the 1PVSO is shown in Table 4.

Table 1. Comparison Between the 2PBVSO and 1PVSO Methods for Solving Problem 1.

TOL	MTD	STEPS	FSTEPS	MAXE	AVEERR	TIME
	1PVSO	598	0	1.22195(1)	3.68458(0)	1518498
10^{-2}	S2PBVSO	364	0	4.40084(2)	2.27716(2)	2063175
	P2PBVSO	364	0	4.40084(2)	2.27716(2)	1887622
	1PVSO	939	0	3.93042(-1)	1.83251(-1)	2376254
10^{-3}	S2PBVSO	349	0	2.27298(1)	5.85638(0)	2100051
	P2PBVSO	349	0	2.27298(1)	5.85638(0)	1807072
	1PVSO	1478	0	3.38966(-2)	1.66894(-2)	3743113
10^{-4}	S2PBVSO	511	0	2.25449(-1)	1.04599(-1)	2849542
	P2PBVSO	511	0	2.25449(-1)	1.04599(-1)	2513270
	1PVSO	2053	0	5.75451(-3)	2.86310(-3)	5209760
10^{-5}	S2PBVSO	961	0	3.20551(-3)	1.57411(-3)	4885463
	P2PBVSO	961	0	3.20551(-3)	1.57411(-3	4343496
	1PVSO	3098	0	5.84631(-4)	2.91622(-4)	7862908
10^{-6}	S2PBVSO	1543	0	5.23500(-4)	2.58489(-4)	7855993
	P2PBVSO	1543	0	5.23500(-4)	2.58489(-4)	7101058
	1PVSO	6648	0	1.08072(-4)	5.39178(-5)	15443732
10^{-7}	S2PBVSO	3515	0	9.80326(-5)	4.87667(-5)	16211339
	P2PBVSO	3515	0	9.80326(-5)	4.87667(-5)	15187703
	1PVSO	11740	0	6.67687(-6)	3.34543(-6)	27334380
10^{-8}	S2PBVSO	5567	0	1.10314(-5)	5.49706(-6)	25699159
	P2PBVSO	5567	0	1.10314(-5)	5.49706(-6)	24060421
	1PVSO	18639	0	6.81149(-7)	3.41212(-7)	43359110
10^{-9}	S2PBVSO	8814	0	1.18653(-6)	5.92066(-7)	40688978
	P2PBVSO	8814	0	1.18653(-6)	5.92066(-7)	37319297
	1PVSO	29576	0	6.86633(-8)	3.44482(-8)	68823756
10^{-10}	S2PBVSO	14550	0	1.01089(-7)	5.04636(-8)	67184711
	P2PBVSO	14550	0	1.01089(-7)	5.04636(-8)	61584492

Table 2. Comparison Between the 2PBVSO and 1PVSO Methods for Solving Problem 2.

TOL	MTD	STEPS	FSTEPS	MAXE	AVEERR	TIME
	1PVSO	30	0	4.31366(-1)	3.24094(-2)	108573
10^{-2}	S2PBVSO	22	0	2.68531(0)	2.10526(-1)	146055
	P2PBVSO	22	0	2.68531(0)	2.10526(-1)	266752
	1PVSO	40	0	7.01467(-2)	3.81878(-3)	141470
10^{-3}	S2PBVSO	29	0	4.58291(-1)	2.59695(-2)	169356
	P2PBVSO	29	0	4.58291(-1)	2.59695(-2)	181533
	1PVSO	55	0	8.76494(-3)	3.68080(-4)	196704
10^{-4}	S2PBVSO	37	0	2.72761(-2)	1.75766(-3)	218415
	P2PBVSO	37	0	2.72761(-2)	1.75766(-3)	232218
	1PVSO	77	0	8.19699(-4)	3.23467(-5)	277540
10^{-5}	S2PBVSO	47	0	2.47936(-3)	9.19042(-5)	280087
	P2PBVSO	47	0	2.47936(-3)	9.19042(-5)	290731
	1PVSO	142	0	1.06305(-4)	6.31511(-6)	504104
10^{-6}	S2PBVSO	65	0	1.46407(-4)	5.31529(-6)	370004
	P2PBVSO	65	0	1.46407(-4)	5.31529(-6)	376870
	1PVSO	211	0	1.42132(-5)	1.01586(-6)	751233
10^{-7}	S2PBVSO	118	0	2.06986(-5)	1.25281(-6)	658933
	P2PBVSO	118	0	2.06986(-5)	1.25281(-6)	657989
	1PVSO	326	0	1.42304(-6)	1.04461(-7)	1163791
10^{-8}	S2PBVSO	177	0	2.00018(-6)	1.31820(-7)	991525
	P2PBVSO	177	0	2.00018(-6)	1.31820(-7)	946544
	1PVSO	506	0	1.42015(-7)	1.06173(-8)	1809574
10^{-9}	S2PBVSO	263	0	2.08704(-7)	1.36801(-8)	1479023
	P2PBVSO	263	0	2.08704(-7)	1.36801(-8)	1435613
	1PVSO	852	0	9.91859(-9)	5.16608(-10)	2885197
10^{-10}	S2PBVSO	413	0	1.92405(-8)	1.28744(-9)	2323099
	P2PBVSO	413	0	1.92405(-8)	1.28744(-9)	2239253
	1PVSO	4245	0	3.71288(-10)	5.38024(-11)	13974180
10^{-11}	S2PBVSO	1927	0	9.48793(-9)	7.90533(-10)	10065314
	P2PBVSO	1927	0	9.48793(-9)	7.90533(-10)	9840786

Table 3. Comparison Between the 2PBVSO and 1PVSO Methods for Solving Problem 3.

TOL	MTD	STEPS	FSTEPS	MAXE	AVEERR	TIME
	1PVSO	82	0	1.44705(-1)	6.47764(-2)	189754
10^{-2}	S2PBVSO	38	0	6.00075(-1)	1.55678(-1)	194799
	P2PBVSO	38	0			309093
	1PVSO	123	0	1.62545(-2)	7.83718(-3)	280473
10^{-3}	S2PBVSO	56	0	1.00808(-1)	3.52228(-2)	243247
	P2PBVSO	56	0	6.00075(-1)	1.55678(-1)	254634
	1PVSO	187	0	1.54921(-3)	7.78556(-4)	428363
10^{-4}	S2PBVSO	100	0	1.30437(-4)	4.47295(-5)	399181
	P2PBVSO	100	0	1.30437(-4)	4.47295(-5)	393886
	1PVSO	289	0	1.28247(-3)	4.74201(-4)	717406
10^{-5}	S2PBVSO	145	0	1.31311(-4)	6.20731(-5)	582361
	P2PBVSO	145	0	1.31311(-4)	6.20731(-5)	584928
	1PVSO	633	0	1.70684(-5)	9.12357(-6)	1326319
10^{-6}	S2PBVSO	219	0	1.48969(-5)	7.40188(-6)	882308
	P2PBVSO	219	0	1.48969(-5)	7.40188(-6)	865647
	1PVSO	1009	0	1.68531(-6)	9.07451(-7)	2116834
10^{-7}	S2PBVSO	487	0	4.46304(-6)	1.42030(-6)	1771549
	P2PBVSO	487	0	4.46304(-6)	1.42030(-6)	1712943
	1PVSO	1605	0	1.67649(-7)	9.09156(-8)	3366259
10^{-8}	S2PBVSO	751	0	3.86601(-7)	1.77181(-7)	2730201
	P2PBVSO	751	0	3.86601(-7)	1.77181(-7)	2719731
	1PVSO	2546	0	1.67497(-8)	9.11330(-90	5340417
10^{-9}	S2PBVSO	1328	0	1.90492(-8)	1.02737(-8)	4835289
	P2PBVSO	1328	0	1.90492(-8)	1.02737(-8)	4791075
	1PVSO	4038	0	1.67310(-9)	9.12964(-10)	8476608
10^{-10}	S2PBVSO	2099	0	1.93675(-9)	1.04958(-9)	7642271
	P2PBVSO	2099	0	1.93675(-9)	1.04958(-9)	7292485
	1PVSO	20056	0	2.79798(-9)	1.05856(-9)	41178823
10^{-11}	S2PBVSO	3366	0	1.82925(-10)	9.94354(-11)	12249378
	P2PBVSO	3366	0	1.82925(-10)	9.94354(-11)	12086923

Table 4. The Ratio Steps and Execution Times of the 2PBVSO Method to the 1PVSO Method.

TOL	MTD	PROB. 1 R.STEP	R.TIME	PROB. 2 R.STEP	R.TIME	PROB. 3 R.STEP	R.TIME
10^{-2}	S2PBVSO	1.64286	0.73600	1.36364	0.74337	2.15790	0.97410
	P2PBVSO	1.64286	0.80445	1.36364	0.40702	2.15790	0.61391.
10^{-3}	S2PBVSO	2.69054	1.13152	1.37931	0.83534	2.19643	1.15304
	P2PBVSO	2.69054	1.31498	1.37931	0.77931	2.19643	1.10148
10^{-4}	S2PBVSO	2.89237	1.31358	1.48649	0.90060	1.87000	1.07310
	P2PBVSO	2.89237	1.48934	1.48649	0.84707	1.87000	1.08753.
10^{-5}	S2PBVSO	2.13632	1.06638	1.63830	0.99091	1.99310	1.23189
	P2PBVSO	2.13632	1.19944	1.63830	0.95463	1.99310	1.22649
10^{-6}	S2PBVSO	2.00778	1.00088	2.18462	1.36243	2.89041	1.50324
	P2PBVSO	2.00778	1.10729	2.18462	1.33761	2.89041	1.53217
10^{-7}	S2PBVSO	1.89132	0.95265	1.78814	1.14008	2.07187	1.19491
	P2PBVSO	1.89132	1.01686	1.78814	1.14171	2.07187	1.23579
10^{-8}	S2PBVSO	2.10886	1.06363	1.84181	1.17374	2.13715	1.23297
	P2PBVSO	2.10886	1.13607	1.84181	1.22952	2.13715	1.23772
10^{-9}	S2PBVSO	2.11470	1.06562	1.92395	1.22349	1.91717	1.10447
	P2PBVSO	2.11470	1.16184	1.92395	1.26049	1.91717	1.11466
10^{-10}	S2PBVSO	2.03272	1.02440	2.06295	1.24196	1.92377	1.10917
	P2PBVSO	2.03272	1.11755	2.06295	1.28846	1.92377	1.16238
10^{-11}	S2PBVSO	-	-	2.20291	1.38835	5.95840	3.36171
	P2PBVSO	-	-	2.20291	1.42003	5.95840	3.40689

8. COMMENTS ON THE RESULTS

The results demonstrate the advantage of using the 2PBVSO method over the 1PVSO method in term of the total number of steps. In all tested problems, it can be observed that the number of steps taken by the former method is always less than the number of steps taken by the latter method for any step size.

It could be observed that the accuracy of the 2-Point and 1-Point method for solving the same test problems is comparable when TOL $\leq 10^{-2}$. The accuracy of the former method is expected to be slightly less than that of the latter method due to the step size of the second point being double and thus reducing the accuracy. However, surprisingly, in some cases the accuracy of the 2-Point

method is better than the 1-Point method.

Both parallel and sequential implementations of the 2-Point method require more time than 1-Point method in Problem 1 when $h = 10^{-2}$. The pattern is reversed as the step size becomes smaller except at $h = 10^{-7}$ where the timing of the former method is slightly more than the latter method. The largest speed up gained by the parallel and sequential 2-Point method occurs at $h = 10^{-4}$ where the ratio number of steps is also the largest.

In Problem 2, the numbers of steps taken by the 2-Point and 1-Point methods are considerably small at tolerances between 10^{-5} to 10^{-2}. It is also observed that the execution times taken by the former method are more than the latter method. This could be justified by the fact that the time spent on performing extra computations required in the 2-point method over-shadowed the advantages of approximating the solution at two points. Moreover, the parallel overheads also appear in the parallel 2-Point method. This explains why it is the slowest in this interval. As expected, the superiority of the 2-Point method begins to show as the number of step increases. This happens when the step size becomes finer with both parallel and sequential implementations of the method enjoy the maximum speed up of 1.42 and 1.39 respectively at the tolerance 10^{-11} where the largest ratio steps also occur. The advantages of 2-Point method are achieved a lot sooner in Problems 3. The method outperform the 1-Point method when the tolerance less than 10^{-2} with the parallel version take the least time as the tolerance becomes finer.

REFERENCES

[1] Henrici P., *Discrete Variable Methods in Ordinary Differential Equations.* New York: John Wiley & Sons, Inc., 1962.

[2] Lambert J.D., *Computational Methods in Ordinary Differential Equations.* New York: John Wiley & Sons, Inc., 1973.

[3] Omar Z.B., and Suleiman M.B., Solving Second Order ODEs Directly Using Parallel 2-Point Explicit Block Method. Prosiding Kolokium Kebangsaan Pengintegrasian Teknologi Dalam Sains Matematik, Universiti Sains Malaysia, 1999; 390-395.

[4] Suleiman M.B., Generalised Muiltistep Adams and Backward Differentiation Methods for the Solution of Stiff and Non-Stiff Ordinary Differential Equations, PhD Thesis, University of Manchester, 1979.

[5] Bronson R., *Schaum's Outline of Modern Introductory Differential Equation*, USA: Mc Graw-Hill, 1973.

[6] Suleiman M.H., Solving Higher Order ODEs Directly by the Direct Integration Method, *Applied Mathematics and Computation,* 1989; **33**(3): 197-219.

[7] Krogh F.T., A Variable Step, Variable Order Multistep Method for the Numerical Solution of Ordinary Differential Equation, Proceedings of the IFIP Congress in Information Processing, 1968; **68**: 194-199.